# Hierarchically Partitioned Implicit Surfaces For Interpolating Large Point Set Models

**Abstract.** We present a novel hierarchical spatial partitioning method for creating interpolating implicit surfaces using compactly supported radial basis functions (RBFs) from scattered surface data. From this hierarchy of functions we can create a range of models from coarse to fine, where a coarse model approximates and a fine model interpolates. Furthermore, our method elegantly handles irregularly sampled data and hole filling because of its multiresolutional approach. Like related methods, we combine neighboring patches without surface discontinuities by overlapping their embedding functions. However, unlike partition-of-unity approaches we do not require an additional explicit blending function to combine patches. Rather, we take advantage of the compact extent of the basis functions to directly solve for each patch's embedding function in a way that does not cause error in neighboring patches. Avoiding overlap error is accomplished by adding phantom constraints to each patch at locations where a neighboring patch has regular constraints within the area of overlap (the function's radius of support). Phantom constraints are also used to ensure the correct results between different levels of the hierarchy. This approach leads to efficient evaluation because we can combine the relevant embedding functions at each point through simple summation. We demonstrate our method on a several very large models including the Thai statue from the Stanford 3D Scanning Repository. Using hierarchical compactly supported RBFs we interpolate all 5 million vertices of the model.

## 1 Introduction

A common problem in computer graphics is interpolating a large set of points on or near a surface to produce a smooth surface. These points may originate as unorganized point sets such as from a 3-D scanning system. They may also come in organized or semiorganized sets from the vertices of polygonal models, which once interpolated can provide a smoother surface than the polygonal one and can be converted to other representations, including a more finely polygonalized one if desired. Such point sets may also come from computer vision analysis of an image or set of images or from interactive modeling tools.

Biological and medical applications represent an important area where such data exist and the smooth reconstruction of surface models resolve specific critical needs. Most biological objects can be assumed to be smooth but often with complex topology. Surface representations in medicine and biology should be manifolds, are often closed, and

should be orientable surfaces with a clear indication of inside and outside. In these respects, implicit surfaces have significant advantages over polygonal meshes and spline patches. Medical applications in particular often involve very large, detailed models and require exact interpolations; the difference between a thin piece of obstructing bone and no bone can be a crucial distinction during a clinical procedure. Additionally, the use of implicit surfaces or any surface model can help create more compact data representations than the existing voxel-based models in medicine.

A number of techniques have emerged for converting such point sets to implicit models that interpolate (or approximate) these points [1–18]. Broadly, we call these *interpolating implicit surfaces*.[1] These methods take the same general approach: known points on the surface define where the implicit surface's embedding function should have a value of 0; known off-surface points, surface normals (either known or fitted), or other assumptions define where the embedding function has nonzero values; and the embedding function is then interpolated using scattered data interpolation techniques such as radial basis functions (RBFs) [1–3, 5–9, 12], (implicit) moving least squares [17], or partition-of-unity blending of local fitting using these or other interpolation methods [10, 13–15]. Though they differ in various ways (the interpolation methods used, the means of defining the non-surface constraints, and the tolerance of fitting the points), they all share this key idea: rather than explicitly interpolating the surface, they interpolate the embedding function implicitly defining the surface.

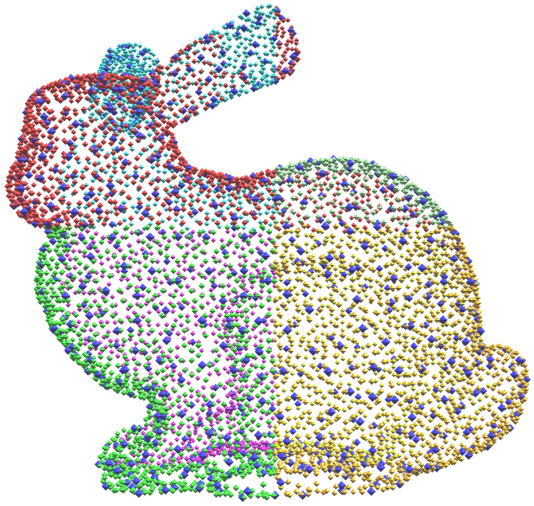

**Fig. 1.** A simple two-level hierarchy of the Stanford bunny. The constraints of the root RBF are dark blue and slightly larger, and each child partition is indicated in different colors.

Many implementations of this idea [1, 3, 8] use thin-plate spline RBFs [19] so as to produce the smoothest interpolation possible. However, the direct formulation of this requires the solving of a large, full, generally ill-conditioned system of equations and quickly becomes computationally impractical for large models. Implementations using other RBFs with infinite support [7, 9, for example] have similar limitations. Various methods have been used to accelerate RBF approaches, including using compactly supported RBF surfaces such as those in [20] to make the required system sparse [5, 11, 12, 16, 21, 22] or approximating a large set of constraints by a well-selected subset [4,

[1] These have also been known in the literature as *variational implicit surfaces*, *implicit surfaces that interpolate*, and *constraint-based implicit surfaces* by various authors.

6]. Others accelerate the surface fitting by subdividing the surface into smaller patches, fitting a surface (or the embedding function for that surface) to each patch, then combining the patches through blending [10, 13–15, 17].

In this paper, we present a novel method for efficiently creating RBF-based implicit representations from the vertices of a polygonal model using hierarchical spatial partitioning as illustrated in Figure 1. Because the resulting surface interpolates the data (to within numerical limits), it is suitable for medical or other high-precision applications. Unlike other approaches that combine local interpolations through compactly supported blending functions, no explicit blending function is required—each level and partitioned patch is calculated so that a simple linear combination of them produces an exact interpolation. We demonstration the interpolation of multi-million point models.

## 2 Related Work and Background

Many techniques have been proposed for organizing point sets into surfaces. Some of these attempt to organize the points into polygonal models [23, 24, and many others]. Others use moving least squares (MLS), defining the surface as fixed points of a nonlinear projection [25–28]. As our goal is to create implicit models, we focus most closely here on such techniques.

Our implementation of compactly-supported RBFs follows most closely that detailed in [5]. This method is based on the general approach of Turk *et al.* [2, 3, 8], which is itself similar to a method first proposed by Savchenko *et al.* [1].

The basic method begins with a set of points known to lie on the desired implicit surface and constrains the interpolated embedding function to have a value of 0 at these points. Using the method of [8], non-zero constraints (often called "normal constraints") are placed at a fixed offset in the direction of the known or desired normals at these surface points. This produces a set of constraints $P = \{(\mathbf{c}_i, h_i)\}$ such that $h_i = 0$ for all $\mathbf{c}_i$ on the surface and $h_i = 1$ for all $\mathbf{c}_i$ at a fixed offset from that surface. An embedding function $f(\mathbf{x})$ is then interpolated from these constraints such that $f(\mathbf{c}_i) = h_i$.

This interpolation is done using an RBF $\phi(r)$ by defining the embedding function $f$ as a weighted sum of these basis functions centered at each of the constraints:

$$f(\mathbf{x}) = \sum_{(\mathbf{c}_i, h_i) \in P} d_i \, \phi(\|\mathbf{x} - \mathbf{c}_i\|) \tag{1}$$

where $d_i$ is the weight of the radial basis function positioned at $\mathbf{c}_i$.[2] To solve for the set of weights $d_i$ that satisfy the known constraints $f(\mathbf{c}_i) = h_i$, substitute each constraint $(\mathbf{c}_i, h_i)$ into Eq. 1:

$$\forall \mathbf{c}_i : f(\mathbf{c}_i) = \sum_{(\mathbf{c}_j, h_j) \in P} d_j \phi(\|\mathbf{c}_i - \mathbf{c}_j\|) = h_i \tag{2}$$

This basic method has been used in graphics for surface fitting from scanned data, interactive shape modeling, and shape interpolation. They have also been applied in computer vision, including using anisotropic basis functions to approximate and smooth

[2] For some RBFs, including the thin-plate spline RBFs, an additional polynomial may also be required.

volumetric representations from 3-D reconstruction algorithms [7, 9], and for representing topology-adaptive active contours [29].

Carr *et al.* [6] extend this technique to large models by using a selectively chosen subset of the model's points so that interpolating the subset approximates the overall model within a specified tolerance. This technique relies on fast evaluation of the embedding function, made possible using fast multipole methods [30]. They also improve the method by adjusting the displacement of the normal constraints to avoid interpenetrating surfaces.

By using compactly supported RBFs, such as those proposed by Wendland [20], one can make this system of equations sparse [5, 11]. By efficiently organizing the points spatially, one can also reduce the time required to compute the system itself. Results presented in [5] show that complexity on the order of $O(n^{1.2-1.5})$ may result, depending on the sparseness of the matrix. As the size of the model increases, one can commensurately reduce the radius of support for the RBFs, thus increasing efficiency while keeping the data density approximately constant. (This method is also similar to that proposed earlier by Muraki's "Blobby Model" [22].)

The primary drawback to using compactly supported radial basis functions alone for surface modeling is that the embedding function is 0 outside one radius of support from the surface. This does not preclude polygonalization, ray-tracing, or many other uses of the surface because it is relatively easy to separate zero sets that result from lack of support. However, it does limit their use for CSG and other operations for which implicit surfaces are useful. The compact support also causes them to fail in areas with low data density, in the limit failing where the surface has holes larger than the support. (See [12] for an excellent discussion of the limitations of compactly supported RBFs for surface modeling, with additional empirical analysis in [21].) These limitations can be overcome using hierarchical, or multilevel, approaches, such as [31, 32] for scatter data interpolation, and [12, 16] for compactly supported RBFs.

Another way to accelerate the surface fitting is to spatially subdivide the surface points into separate patches, then interpolate (or approximate) each patch and blend the results using partition-of-unity blending. By blending local approximations instead of directly trying to fit the entire model at once, this method provides efficient processing for very large models. Wendland [10] first proposed combining RBF interpolation of "mildly overlapping" domains with partition-of-unity blending of the resulting local interpolations to produce a global solution. Ohtake *et al.* [13] applied this idea to implicit surface fitting but used least-squares fitting of each surface patch, recursively subdividing each patch until the approximation is within desired tolerances. Tobor *et al.* [14, 15] also applied recursive subdivision and partition-of-unity blending to implicit surface fitting, using an approach closer to that proposed by Wendland.

Shen *et al.* [17] use a similar method to blend local embedding functions. In their case, they begin with a potentially unorganized polygonal model and define a local embedding function on a per-polygon basis using the polygon normals to avoid the need for additional normal constraint points. These local embedding functions are blended together using moving least squares, which allows them to perform either exact interpolation or approximation depending on the choice of weighting function. (They point

out that their technique differs from the MLS approaches in [25, 28] and related work, so they call theirs an implicit MLS approach.)

Key to partition-of-unity or moving-least-squares methods is the use of a compactly supported weighting function to blend separate patches or the effects of individual points in a neighborhood. We demonstrate a new method for creating and blending interpolations for separate patches that uses compactly supported RBFs to interpolate the patches and, due to their compactly supported nature, does not require the use of a separate explicit blending function.

## 3 Method

Building a single embedding function that interpolates all the points of a very large data set is not feasible. Therefore, we take a coarse-to-fine, top-down approach to partition the problem into smaller, tractable embedding functions. Our method builds a large-scale embedding function, and then successively refines it with smaller-scale incremental functions. The two main components of our method are selecting the points for each node in the hierarchy and creating phantom constraints to allow overlapping embedding functions. Using phantom constraints to clamp each embedding function allows us to combine them simply by addition rather than requiring a blending function.

### 3.1 Building a Hierarchy

To build a hierarchy we use an octree to span the input points, which is traversed from the top down. Points are first selected for the root, producing an embedding function for a base model. Then points for the each of the children of the root are selected, adding detail at a finer resolution. After solving the refining embedding functions for the eight children, we proceed to the grandchildren, and so on. When building any given node, the functions for the nodes above it have been solved already.

**Selecting Points For a Node** Points in a node's octant are selected based on a random Poisson-disk distribution [33]. However, the traditional Poisson-disk distribution, where there is a minimum Euclidean distance between any two points, tends to undersample regions of high curvature. We would like to allow sample points to be closer together in high curvature regions.

Comparing the normal directions of nearby points is an efficient estimate of local curvature. A region with points that have disparate normals requires a higher sampling rate. To achieve adaptive sampling we use a modified distance function based on the points' normals. If two points have identical normals, the distance between them is the same as the Euclidean distance. However, if their normals differ we would like them to appear to be farther apart. The net effect is to place samples closer together.

We use a modified distance function that scales the Euclidean distance by a quadratic function of the angle $\theta$ between the normal vectors:

$$f(\theta) = \frac{1}{2}\cos(\theta)^2 - \frac{7}{2}\cos(\theta) + 4 \tag{3}$$

This function is 1 when the angle is 0, i.e., the two normals are aligned. It is 4 when the normals are perpendicular and 8 when the normals are directly opposed. E.g, $f(0) = 1$, $f(\frac{\pi}{2}) = 4$ and $f(\pi) = 8$.

So the net distance function is

$$\text{dist}(\mathbf{x}_1, \mathbf{x}_2, \theta) = \|\mathbf{x}_1 - \mathbf{x}_2\| \left( \tfrac{1}{2}\cos(\theta)^2 - \tfrac{7}{2}\cos(\theta) + 4 \right) \tag{4}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the two points and $\theta$ is angle between their normals.

Another possible distance function would be the geodesic distance on a surface, if a surface mesh were available. Using the geodesic distance would help problems that occurs when surfaces are very close together but not directly connected. However in small regions of high curvature the geodesic distance would still not place enough samples and is expensive to compute.

**Selecting Points For the Root** Selecting points for the root embedding function is especially important because error in the root propagates throughout the hierarchy. The more error there is in a parent node, the more "energy" required at a child node to bend the embedding function to fit. Since the root node affects all other nodes, we are more particular in selecting its points.

At the root node, in addition to the modified Poisson-disk distribution mentioned earlier, we attempt to select points that are "representative" of a local region. The idea is to pick points that capture the larger-scale shape of a region, pushing smaller scale detail or noise to nodes lower in the hierarchy.

For the root node candidate points are screened by comparing its normal direction with the normals of points around it. If the point's normal is too disparate from those of its neighbors, it is not selected. Specifically the average dot product of a candidate point's normal with the normals around it is computed. If it is below a specific threshold (0.1), the point is not selected.

Figure 1 shows a two-level hierarchy of RBFs of the Stanford bunny. The hierarchy consists of a root RBF and eight children. The constraints are colored by node. The root's constraints are dark blue and slightly larger.

**Embedding Function For a Node** As mentioned previously, the hierarchy is built from the top down. Once points have been selected for a node, interior, surface, and exterior constraints are placed for each point. The embedding function also requires a level set value for each constraint and a radius of support for the compact RBF.

The root embedding function should produce the correct results at the locations of the root's constraints. Therefore the constraint values are determined solely by the type of constraint. By default at a surface constraint the function should be 0, at an interior constraint it is should be 1 and at an exterior constraint it should be $-1$.

Using the notation of Eqs. 1 and 2, we can write the root embedding function $f_0$ defined by the set of root constraints $P_0 = \{(\mathbf{c}_i, h_i)\}$ using root-level RBF $\phi_0(r)$ as follows:

$$f_0(\mathbf{x}) = \sum_{(\mathbf{c}_i, h_i) \in P_0} d_{0i}\, \phi_0(\|\mathbf{x} - \mathbf{c}_i\|) \tag{5}$$

where the root-level weights $d_{0i}$ are determined by solving the system of equations

$$\forall \mathbf{c}_i \in P_0 : f_0(\mathbf{c}_i) = h_i \tag{6}$$

The embedding function of a child node is an increment that corrects the parent function at the location of the child node's constraints. For example at a child node's surface constraint the net function should to evaluate to 0. However, the parent function evaluates to some value $\alpha$. Therefore the child's function should evaluate to $-\alpha$ to correct the error. Thus at each child constraint location, the hierarchy of embedding functions above the child node is evaluated, and a value is given to the child constraint that corrects the result of the nodes above it.

Thus, we may write a single child level's embedding function $f_1$ defined by the child node's constraints $P_1 = \{(\mathbf{c}_i, h_i)\}$ and child-level RBF $\phi_1(r)$, along with the parent node's constraints $P_0$ and embedding function $f_0$, as follows:

$$f_1(\mathbf{x}) = \sum_{(\mathbf{c}_i,h_i)\in P_0 \cup P_1} d_{1i}\, \phi_1(\|\mathbf{x}-\mathbf{c}_i\|) \tag{7}$$

where the child-level weights $d_{1i}$ are determined by solving the system of equations

$$\forall \mathbf{c}_i \in P_0 \cup P_1 : f_1(\mathbf{c}_i) = h_i - f_0(\mathbf{c}_i) \tag{8}$$

Note that each level $k$ of the hierarchy uses its own RBF $\phi_k(r)$ and weights $d_{ki}$.

Since Eq. 6 already holds for the root nodes, the root embedding function $f_0$ already evaluates correctly at the root nodes and no correction is required by the child embedding function:

$$\forall \mathbf{c}_i \in P_0 : f_1(\mathbf{c}_i) = 0 \tag{9}$$

This process may be continued to additional levels of the hierarchy.

Note that this formulation includes only a single node at each level of the hierarchy. Solving for and combining embedding functions for multiple nodes at each level of the hierarchy is addressed in Section 3.2.

Once all the constraint values for a node have been determined, the radius of support for the compact RBF for that node must be determined. We attempt to keep the same number of points per node, and nodes at different levels in the hierarchy cover different sized regions. Naturally different-level nodes should have compact RBFs with different radii. In our approach the user selects the radius for the root, and each descendant is given a radius proportional to that root radius and to its own size.

Typically compact RBFs are used for the embedding functions of all of the nodes in the hierarchy, but using only compactly supported RBFs have the problem of the function being undefined in some regions. Any location that is outside of all constraints' radii of support will not have a defined embedding function. Therefore we also allow the option of using a thin plate spline RBF for the root node (see Figure 3), eliminating the problem. The downside is that it is much more expensive to solve and evaluate, since all the constraints affect each other and all other points in space [5]. But since the embedding function for the root node uses only a limited subset of points, it is still practical even for otherwise large models.

### 3.2 Phantom Constraints

Managing overlapping embedding functions is a common problem that occurs when attempting to partition a point set. To interpolate all the points in a data set, we must guarantee that the combination of all embedding functions that impinge on a point produces the exact value we require. Our task is simplified by the compact RBF's limited extent. Therefore at any given point only relatively few embedding functions need to be combined and evaluated.

Our approach is to place *phantom constraints* in a given node to clamp its embedding function. Phantom constraints are placed in regions where nodes overlap and where we want to suppress the influence of the node's embedding function. In this way, phantom constraints serve much the same purpose as the blending function in partition-of-unity approaches but without explicit blending during evaluation of the implicit surface's embedding function. The locations for phantom constraints fall into two categories: locations that have been inherited from regular constraints in ancestral nodes, and locations from regular constraints in adjacent sibling nodes. Using a top-down approach means constraint locations from descendant nodes can be ignored.

A child node's embedding function is an incremental change applied to the sum of its ancestor embedding functions as mentioned in Section 3.1. For our purposes an ancestor node is any node above a given node that overlaps with the node, not just direct ancestors. Since a child embedding function is an increment to the functions above it, a regular constraint of the child is given a value that corrects the summed ancestor functions. In addition to moving the net embedding function towards a child's regular constraint locations, we need to ensure that the child's embedding function does not incorrectly move the net function at all its ancestors' constraint locations. Therefore, phantom constraints that have values of 0 are placed in the child RBF at all ancestor constraint locations within the bounds of the child.

Similarly a node should not be incorrectly affected by neighboring sibling nodes. Therefore, the regular constraints of any neighboring sibling that might affect a node become corresponding phantom constraints for the node. These neighboring regular constraints are any that fall within the bounds of the node's regular constraints augmented by its radius of support.

Extending the notation of Eqs. 5–9, we define $P_{lk} = \{(\mathbf{c}_i, h_i)\}$ as the set of constraints for node $k$ of level $l$. We also define $\hat{P}_{lk} = \{(\mathbf{c}_i, h_i)\}$ as the set of phantom constraints relevant to this node and the function $\hat{f}_{lk}$ as the sum of all other embedding functions relevant to this node (i.e., those higher up in the hierarchy whose support-expanded regions overlap this node's support-expanded region). We may thus write the embedding function for this child node in terms of these constraints and the node's RBF $\phi_{lk}(r)$ as

$$f_{lk}(\mathbf{x}) = \sum_{(\mathbf{c}_i, h_i) \in P_{lk} \cup \hat{P}_{lk}} d_{lki}\, \phi_{lk}(\|\mathbf{x} - \mathbf{c}_i\|) \tag{10}$$

where the child node's weights $d_{lki}$ are determined by solving the system of equations

$$\forall \mathbf{c}_i \in P_{lk} \cup \hat{P}_{lk} : f_{lk}(\mathbf{c}_i) = h_i - \hat{f}_{lk}(\mathbf{c}_i) \tag{11}$$

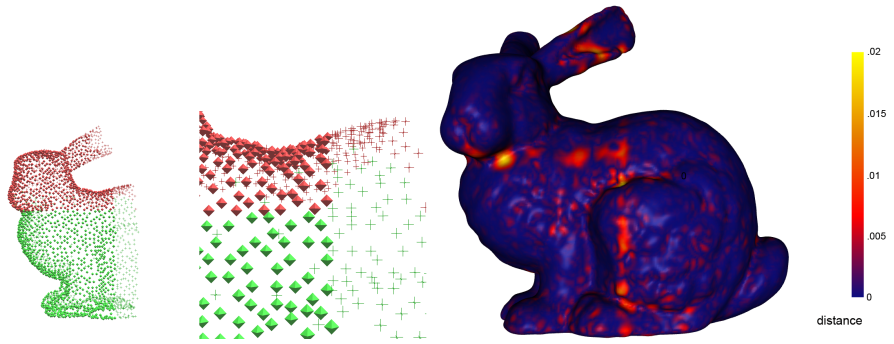


**Fig. 2.** Phantom constraints. a) the constraints of two neighboring child nodes of the bunny. The octahedra are regular constraints, and the crosses are phantom constraints. b) the effects of phantom constraints on the embedding function. The left side of the bunny does not have phantom constraints from neighboring nodes, while the right side does. The color shows the distance error between the embedding function and the original surface.

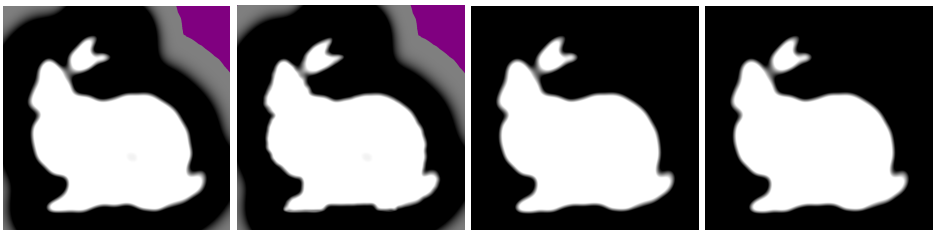


**Fig. 3.** Slices through the embedding functions. a) a compact RBF root. b) a two level hierarchy, with a compact root. c) a thin plate spline root. d) a two level hierarchy, with a thin plate spline root. The purple is where the compact RBF is undefined.

Again, this node's embedding function provides incremental refinement only and does not change the result at the phantom constraints from other nodes:

$$\forall \mathbf{c}_i \in \hat{P}_{lk} : f_{lk}(\mathbf{c}_i) = 0 \tag{12}$$

Figure 2a is two overlapping child nodes of the bunny. The octahedra represent regular constraints, while the crosses represent phantom constraints. The phantom constraints contained with the bounds of a node's octant have been inherited from the root node, while those outside the octant come from neighboring sibling nodes.

Figure 2b illustrates the error that can occur from overlapping embedding functions that do not have phantom constraints. On the left side of the bunny there are no phantom constraints from neighboring octants. On the right side there are phantom constraints. The surface is colored by the distance error between the embedding function and the original surface mesh. Clearly there is much more error on the left side, particularly where octants abut. Also the error bleeds into the right side of the bunny because the functions on the left are not evaluating to 0 to the right.

Figure 3 shows slices through four embedding function of the bunny. Images 3a and 3b use a compact RBF for the root node, while 3c and 3d use a thin plate spline at the root. In the left pair the purple is a region where the compact RBF is not defined. Images 3a and 3c are slices through the embedding functions of just the root nodes. The images that slice through two level hierarchies (3b and 3d) clearly show sharper boundaries and more detail. For instance, the bottoms of the bunny are less rounded.

Adding phantom constraints outside of a given node's bounds expands the region of space where the node's RBF must be evaluated. This region is the union of spheres centered at each constraint where all the spheres have a radius that is the compact RBF's radius of support. However, this expansion can be nullified by only defining the embedding function in the original region defined by the regular constraints. For any location that has only phantom constraints within the radius, we make the embedding function return 0, since the purpose of phantom constraints is to suppress the function.

## 4 Results

To demonstrate the efficacy of our method, we applied it to three large data sets: a skull, a dragon and a statue. The hierarchies of these models are described in Section 4.1. From these data sets Section 4.2 analyzes the statistics and error characteristics of our method. Section 4.3 provides some implementation details.

### 4.1 Examples

The first example data set is a CT scan of a dry skull, provided by GE Medical Research. From the original 571,794 vertices we built an implicit hierarchy of 1.8 million constraints in a octree of height 3. In all our examples we used surface and exterior constraints. Figure 4 shows the constraints of the leaf nodes above and an iso-surface extracted from our embedding function below. Only the leaf node constraints are displayed for visual clarity. Apparently empty patches occur because those leaf nodes have few constraints. In those cases most of the constraints are higher in the hierarchy.

The second data set is the Asian Dragon from Stanford's 3D Scanning Repository. Derived from the data set's 3.6 million vertices, our implicit model has 9.1 million constraints in a 5 level octree. Figure 5 shows the constraints of the leaf nodes on the left and an extracted iso-surface on the right. Figure 6 shows surfaces extracted from each of the levels of the implicit hierarchy. The upper left image is of the original mesh. The upper middle image is of the root node. The upper right

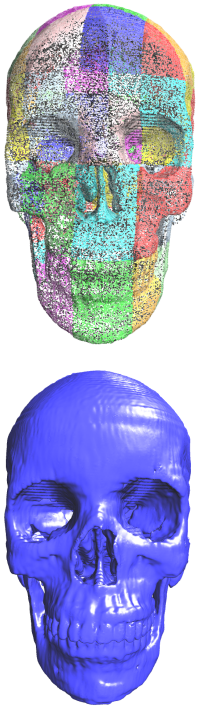

**Fig. 4.** CT scan of a skull, by GE. On top are constraints of the leaf nodes of the skull's implicit hierarchy, and below is an iso-surface extracted from the embedding function.

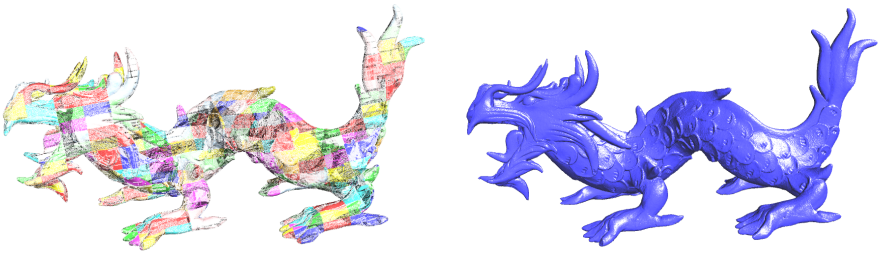

**Fig. 5.** Stanford's Asian Dragon. On the left are the constraints of the leaf nodes of the implicit hierarchy, and on the right is an iso-surface extracted from the embedding function.

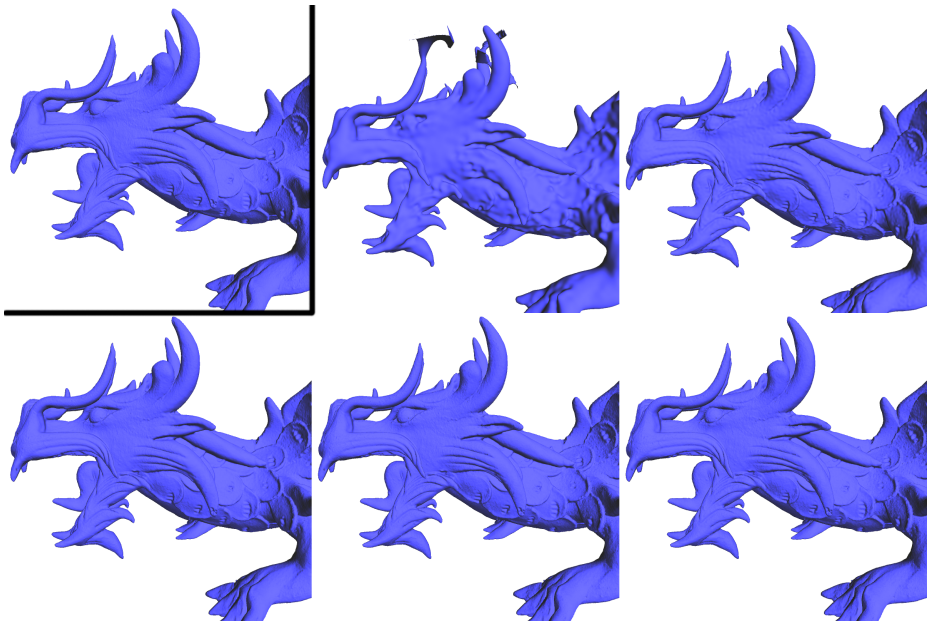

**Fig. 6.** Surfaces from levels of the dragon's hierarchy. In the upper left is the head of the original dragon. From left to right and top to bottom are surfaces extracted from the 5 hierarchy levels.

image and lower three images are of each successive level. The lower three images are essentially indistinguishable from the original. The root node image (upper middle) shows spurious surfaces shooting off form the horns. These problems are typical of compact RBFs in undersampled regions of higher curvature.

The third data set is the Thai Statue, also from Stanford's 3D Scanning Repository. The original model consists of 5 million vertices. To those we added 426,245 on the bottom of the statue, which was not scanned. Our implicit model has 17.5 million constraints in a 5 level octree. Figure 7 shows the five levels in the hierarchy, starting with the root on the left, and an iso-surface extracted from the embedding function.

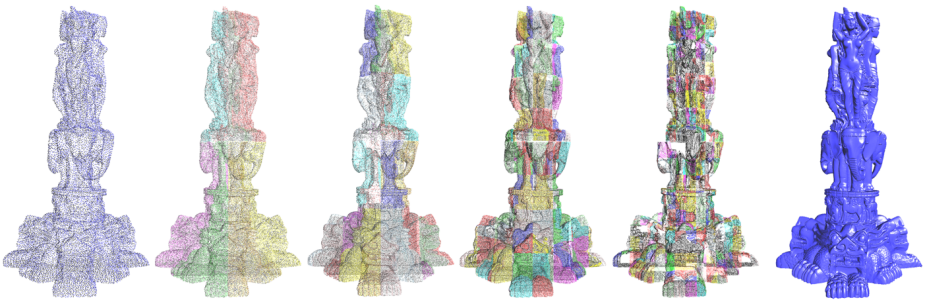

**Fig. 7.** Stanford's Thai statue. The 5 levels of the hierarchy of constraints and an extracted iso-surface. The hierarchy contains 17.5 million constraints.

### 4.2 Statistics

Table 1 shows statistics for the three data sets. The upper section of the table shows statistics for each entire model, and the middle section shows statistical averages per implicit evaluation. The bottom section shows the error in each implicit surface. To compute the statistics in the middle and bottom sections, the embedding function was evaluated at every surface constraint location in each data set.

The implicit error of a constraint is the unsigned difference between the value a constraint should have and the value returned by the embedding function. By default, surface constraints should have a value of 0.

The distance error is the distance between a surface constraint's location and a root (zero) of the embedding function. The root was found by searching the embedding function along the constraint's normal direction. The data sets had extents of 374.6, 201.3, 395.9 along their longest axes respectively, and the exterior constraints were offset by a distance of $10^{-4}$ in all cases. Values of 0 for surface constraints and $-1$ for exterior constraints of the embedding function results in gradients on the order of $10^5$. Thus one would expect implicit errors on the order of $10^5$ times greater than the distance errors, and the statistics bear this expectation out.

In our examples adding phantom constraints increases the number of constraints in the data sets by an average of 51.7% so that phantom constraints make up 31.6% of the constraints. However, since the phantom constraints tend to occur towards the bottom of the hierarchy, i.e. in the nodes with smaller extents, the average number of phantom constraints encountered per function evaluation is lower. In all they represent only 12.7% of the constraints when evaluating the embedding function.

In addition to measuring the error in the implicit function, we measured the distance error between the original meshes and iso-surfaces extracted from different levels of the implicit hierarchy. Figure 8 shows the results in a log chart. A stochastic symmetric difference method was used to determine the average distance between each extracted iso-surface and its original surface, in units of the original surface. The curves demonstrate how the addition of levels in hierarchy improves the accuracy of the iso-surface.

| | | CT skull | Asian dragon | Thai statue |
|---|---|---|---|---|
| Hierarchy statistics | *vertices* | 571,794 | 3,609,455 | 4,999,996 |
| | *regular constraints* | 1,143,586 | 7,218,836 | 10,852,316 |
| | *phantom constraints* | 661,619 | 1,867,784 | 6,632,801 |
| | *total constraints* | 1,825,205 | 9,086,620 | 17,485,117 |
| | *tree nodes* | 63 | 744 | 799 |
| | *tree height* | 3 | 5 | 5 |
| | *build time (minutes)* | 7.10 | 38.13 | 127.35 |
| Averages per evaluation | *regular constraints* | 207.66 | 630.31 | 961.77 |
| | *phantom constraints* | 18.98 | 89.75 | 202.00 |
| | *number of nodes* | 3.34 | 5.84 | 6.59 |
| Error | *avg. implicit error* | $7.400 \times 10^{-08}$ | $2.543 \times 10^{-07}$ | $1.498 \times 10^{-06}$ |
| | *max. implicit error* | $4.176 \times 10^{-05}$ | $1.651 \times 10^{-05}$ | $1.495 \times 10^{-04}$ |
| | *avg. distance error* | $9.189 \times 10^{-13}$ | $7.239 \times 10^{-12}$ | $2.123 \times 10^{-10}$ |
| | *max. distance error* | $2.205 \times 10^{-08}$ | $2.281 \times 10^{-09}$ | $2.270 \times 10^{-08}$ |

**Table 1.** Statistics for the hierarchical implicit surfaces for the three example data sets.

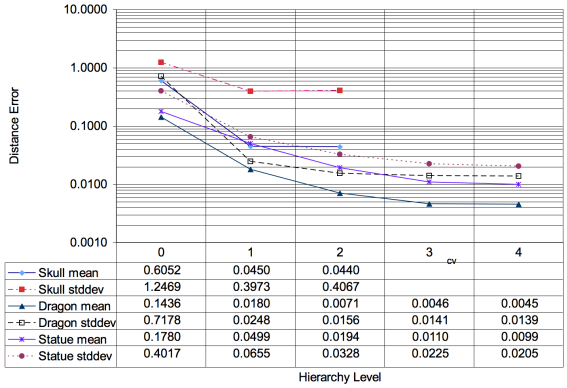


**Fig. 8.** Error statistics for iso-surfaces extracted from different levels of the implicit hierarchy. Each iso-surface was measured against the original meshes.

### 4.3 Implementation

The example implicit hierarchies were built on a SGI Altix system with four 1.4 GHz Itanium 2 processors and 8 GB of main memory. The most time consuming sections of code, solving each node's sparse matrix and computing all the constraints' values, were parallelized. Computing a constraint's value is required for non-root nodes, since its value depends on the embedding functions above it in the hierarchy. The matrices were solved using the LDL solver in SGI's Scientific Computing Software Library.

## 5 Conclusions

We have developed a technique for generating implicit surfaces from large point sets. This method employs a hierarchical spatial partitioning that imposes a successive series of embedding functions that are constrained so that when they are added to one another, they interpolate the point set. Our approach begins with the careful selection of a representative subset of the point set from which an interpolating implicit surface that provides a basic model can be created using linear combinations of compactly supported radial basis functions. This base model interpolates the core subset of data points

and serves as the foundation for the coarse-to-fine hierarchy. The data space is recursively divided into an octree with additional data points selected, and more detailed embedding functions are derived for each child octant that, when added to the base model, accurately interpolate the more complete, higher resolution model. Neighboring spatial partitions are supplemented with overlapping points, phantom constraints, that assure smooth transitions between adjoining embedding functions. No additional blending functions are required because our compactly supported radial basis functions have a limited radius of influence, imposing a predictable margin between partitions and gradual diminishing of effect between them. Furthermore, our method elegantly handles irregularly sampled data and hole filling because of its multiresolutional approach.

Future work in this area includes exploring new criteria for selecting representative points for the base model and the detailed, higher resolution embedding functions. Improved measurements for surface curvature will lead to efficient computation of implicit surfaces. New measures of the saliency of critical elements of the point set may include the detection of higher order features derived from other differential geometric measurements including parabolic curves denoting inflections in Gaussian curvature. In the future, even larger models may be accommodated by developing out-of-core methods for performing the necessary linear algebraic computations. Adjustable, adaptive spatial partitioning may also help to process very large models.

## References


1. Savchenko, V.V., Pasko, A.A., Okunev, O.G., Kunii, T.L.: Function representation of solids reconstructed from scattered surface points and contours. Computer Graphics Forum **14**(4) (1995) 181–188
2. Turk, G., O'Brien, J.F.: Variational implicit surfaces. Technical Report GIT-GVU-99-15, Georgia Institute of Technology (1998)
3. Turk, G., O'Brien, J.F.: Shape transformation using variational implicit functions. Computer Graphics **33**(Annual Conference Series) (1999) 335–342
4. Yngve, G., Turk, G.: Creating smooth implicit surfaces from polygonal meshes. Technical Report GIT-GVU-99-42, Georgia Institute of Technology (1999)
5. Morse, B.S., Yoo, T.S., Rheingans, P., Chen, D.T., Subramanian, K.R.: Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In: Shape Modeling International 2001, Genoa, Italy (2001) 89–98
6. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3D objects with radial basis functions. In: Proceedings of SIGGRAPH 2001. (2001) 67–76
7. Dinh, H.Q., Turk, G., Slabaugh, G.: Reconstructing surfaces using anisotropic basis functions. In: Proc. Eighth International Conference on Computer Vision (ICCV 2001). (2001)
8. Turk, G., O'Brien, J.F.: Modelling with implicit surfaces that interpolate. ACM Transactions on Graphics **21**(4) (2002) 855–873
9. Dinh, H., Turk, G., Slabaugh, G.: Reconstructing surfaces by volumetric regularization using radial basis functions. IEEE Trans. on Pattern Analysis and Machine Intelligence (2002)
10. Wendland, H.: Fast evaluation of radial basis functions: Methods based on partition of unity. In Chui, C.K., Schumaker, L.L., Stöckler, J., eds.: Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt University Press, Nashville, TN (2002) 472–483
11. Kojekine, N., Hagiwara, I., Savchenko, V.: Software tools using CSRBFs for processing scattered data. Computers & Graphics **27**(2) (2003) 311–319

12. Ohtake, Y., Belyaev, A., Seidel, H.: A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. In: Shape Modeling International 2003. (2003)
13. Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., Seidel, H.: Multi-level partition of unity implicits. ACM TOG (Proc. SIGGRAPH 2003) **22**(3) (2003) 463–470
14. Tobor, I., Reuter, P., Schlick, C.: Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions. In: WSCG (Winter School of Computer Graphics). (2004)
15. Tobor, I., Reuter, P., Schlick, C.: Multiresolution reconstruction of implicit surfaces with attributes from large unorganized point sets. In: Proceedings of Shape Modeling International (SMI 2004). (2004) 19–30
16. Ohtake, Y., Belyaev, A., Seidel, H.P.: 3d scattered data approximation with adaptive compactly supported radial basis functions. In: Shape Modeling International 2004. (2004)
17. Shen, C., O'Brien, J.F., Shewchuk, J.R.: Interpolating and approximating implicit surfaces from polygon soup. In: Proceedings of ACM SIGGRAPH 2004, ACM Press (2004) 896–904
18. Nielson, G.M.: Radial hermite operators for scattered point cloud data with normal vectors and applications to implicitizing polygon mesh surfaces for generalized CSG operations and smoothing. In: 15th IEEE Visualization 2004 (VIS'04). (2004) 203–210
19. Duchon, J.: Sur l'erruer d'interpolation des fonctions de plusieurs variables par les $d^m$ splines. R.A.I.R.O Analyse numerique **12**(4) (1978) 325–334
20. Wendland, H.: Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. AICM **4** (1995) 389–396
21. Morse, B., Liu, W., Otis, L.: Empirical analysis of computational and accuracy tradeoffs using compactly supported radial basis functions for surface reconstruction. In: Proceedings Shape Modeling International (SMI'04). (2004) 358–361
22. Muraki, S.: Volumetric shape description of range data using "blobby model". In: Proceedings of ACM SIGGRAPH 1991. Computer Graphics Proceedings, Annual Conference Series, ACM Press / ACM SIGGRAPH (1991) 227–235
23. Amenta, N., Bern, M., Kamvysselis, M.: A new Voronoi-based surface reconstruction algorithm. In: Proceedings of SIGGRAPH 98. Volume 32. (1998) 415–421
24. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: Proceedings of SIGGRAPH 92. Volume 26. (1992) 71–78
25. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Point set surfaces. IEEE Visualization 2001 (2001) 21–28
26. Amenta, N., Kil, Y.J.: Defining point-set surfaces. ACM Trans. on Graphics **23**(3) (2004) 264–270
27. Fleishman, S., Alexa, M., Cohen-Or, D., Silva, C.T.: Progressive point set surfaces. ACM Transactions on Graphics **22** (2003)
28. Levin, D.: Mesh-independent surface interpolation. In Brunnet, G., Hamann, B., Mueller, K., Linsen, L., eds.: Geometric Modeling for Scientific Visualization. Springer-Verlag (2003)
29. Morse, B., Liu, W., Yoo, T., Subramanian, K.R.: Active contours using a constraint-based implicit representation. In: Proc. Computer Vision and Pattern Recognition (CVPR). (2005)
30. Beatson, R.K., Newsam, G.N.: Fast evaluation of radial basis functions. Comput. Math. Appl. **24** (1992) 7–19
31. Floater, M., Iske, A.: Multistep scattered data interpolation using compactly supported radial basis functions. Journal of Comp. Appl. Math. **73** (1996) 65–78
32. Iske, A., Levesley, J.: Multilevel scattered data approximation by adaptive domain decomposition. In: Numerical Algorithms. Volume 39. (2005) 187–198
33. Cook, R.L.: Stochastic sampling in computer graphics. ACM Trans. Graph. **5**(1) (1986) 51–72